PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re U.S. Patent No. 7,156,665	)	Serial No. 09/868,686
	)	
Inventor(s): Martha Torrey O'CONNOR *et al*	)	Filed: September 4, 2001
	)	
Issue Date: January 2, 2007	)	Attorney Docket No. 005222.00157

For: GOAL BASED EDUCATION SYSTEM WITH SUPPORT FOR DYNAMIC TAILORED FEEDBACK

REQUEST FOR CERTIFICATE OF CORRECTION

U.S. Patent and Trademark Office
Customer Service Window
Randolph Building, Mail Stop: Certificate of Correction Branch
401 Dulany Street
Alexandria, VA 22314

Sir:

Pursuant to 35 U.S.C. § 254 and 37 C.F.R. § 1.323, this is a request for the issuance of a Certificate of Correction in the above-identified patent. Two (2) copies of PTO Form 1050 are appended. The complete Certificate of Correction involves one page.

One of the mistakes identified in the appended Form occurred through no fault of the Patent and Trademark Office, as disclosed by the records of the application, which matured into this patent. Enclosed for your convenience are the relevant sections of the International Publication WO 00/38144, submitted June 20, 2001. These errors occurred in good faith without deceptive intent.

Issuance of the Certificate of Correction containing the corrections is earnestly requested. Please charge the requisite fee of $100.00, and any additional fee, which may be associated to our Deposit Account No. 19-0733.

Respectfully submitted,

BANNER & WITCOFF, LTD.

Dated: February 28, 2007

By: Kenneth F. Smolik
Kenneth F. Smolik
Reg. No. 44,344

1001 G Street, N.W. (11th Fl.)
Washington, D.C. 20001
(202) 824-3000

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.: 7,156,665

DATED: January 2, 2007

INVENTOR(S): Martha Torrey O'CONNOR *et al*

It is certified that errors appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In the Specification, Column 17, Line 1:
Please replace "QVIDctrl.cis" with --QVIDctrl.cls--

In the Specification, Column 23, Line 52:
Please replace "Oust" with --(just--

In the Specification, Column 40, Line 67:
Please replace "ICA" with --ICA.--

In the Specification, Column 42, Line 41:
Please replace "boalean" with --boolean--

In the Specification, Column 42, Line 43:
Please replace "ong" with --long--

Mailing Address of Sender:

Banner & Witcoff, Ltd.
11th Floor
1001 G Street, N.W.
Washington, DC 20001-4597

No. of add'l copies
@ $0.50 per page

FORM PTO 1050 (Rev.2-93)

FORM PTO-1390 (REV. 11-2000) U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE

TRANSMITTAL LETTER TO THE UNITED STATES DESIGNATED/ELECTED OFFICE (DO/EO/US) CONCERNING A FILING UNDER 35 U.S.C. 371

ATTORNEY'S DOCKET NUMBER: 05222.00157

U.S. APPLICATION NO. (If known, see 37 CFR 1.5): UNASSIGNED 09/868686

INTERNATIONAL APPLICATION NO.	INTERNATIONAL FILING DATE	PRIORITY DATE CLAIMED
PCT/US99/02726	8 February 1999	22 December 1998

TITLE OF INVENTION: A SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR A GOAL BASED EDUCATIONAL SYSTEM WITH SUPPORT FOR DYNAMIC TAILORED FEEDBACK

APPLICANT(S) FOR DO/EO/US: O'CONNOR, et al.

Applicant herewith submits to the United States Designated/Elected Office (DO/EO/US) the following items and other information:

1. [x] This is a **FIRST** submission of items concerning a filing under 35 U.S.C. 371.
2. [ ] This is a **SECOND** or **SUBSEQUENT** submission of items concerning a filing under 35 U.S.C. 371.
3. [ ] This is an express request to begin national examination procedures (35 U.S.C. 371(f)). The submission must include items (5), (6), (9) and (21) indicated below.
4. [X] The US has been elected by the expiration of 19 months from the priority date (Article 31).
5. [x] A copy of the International Application as filed (35 U.S.C. 371(c)(2))
   a. [ ] is attached hereto (required only if not communicated by the International Bureau).
   b. [x] has been communicated by the International Bureau.
   c. [x] is not required, as the application was filed in the United States Receiving Office (RO/US).
6. [ ] An English language translation of the International Application as filed (35 U.S.C. 371(c)(2)).
   a. [ ] is attached hereto.
   b. [ ] has been previously submitted under 35 U.S.C. 154(d)(4).
7. [ ] Amendments to the claims of the International Aplication under PCT Article 19 (35 U.S.C. 371(c)(3))
   a. [ ] are attached hereto (required only if not communicated by the International Bureau).
   b. [ ] have been communicated by the International Bureau.
   c. [ ] have not been made; however, the time limit for making such amendments has NOT expired.
   d. [ ] have not been made and will not be made.
8. [ ] An English language translation of the amendments to the claims under PCT Article 19 (35 U.S.C. 371 (c)(3)).
9. [ ] An oath or declaration of the inventor(s) (35 U.S.C. 371(c)(4)).
10. [ ] An English lanugage translation of the annexes of the International Preliminary Examination Report under PCT Article 36 (35 U.S.C. 371(c)(5)).

Items 11 to 20 below concern document(s) or information included:

11. [ ] An Information Disclosure Statement under 37 CFR 1.97 and 1.98.
12. [ ] An assignment document for recording. A separate cover sheet in compliance with 37 CFR 3.28 and 3.31 is included.
13. [ ] A FIRST preliminary amendment.
14. [ ] A SECOND or SUBSEQUENT preliminary amendment.
15. [ ] A substitute specification.
16. [ ] A change of power of attorney and/or address letter.
17. [ ] A computer-readable form of the sequence listing in accordance with PCT Rule 13ter.2 and 35 U.S.C. 1.821 - 1.825.
18. [ ] A second copy of the published international application under 35 U.S.C. 154(d)(4).
19. [ ] A second copy of the English language translation of the international application under 35 U.S.C. 154(d)(4).
20. [x] Other items or information:

International Publication WO 00/38144

1 Form PCT/IB/306
Request for Recording a Change Under Rule 92Bis

structure that is inter-linked with the other directories. This structure must be maintained to assure consistency and compatibility between projects to clarify project differences, and architecture updates.

The _Arch directory stores many of the most common parts of the system architecture. These files generally do not change and can be reused in any area of the project. If there is common visual basic code for applications that will continuously be used in other applications, the files will be housed in a folder in this directory. The sub-directories in the _Arch directory are broken into certain objects of the main project. Object in this case refers to parts of a project that are commonly referred to within the project. For example, modules and classes are defined here, and the directory is analogous to a library of functions, APIs, etc... that do not change. For example the IcaObj directory stores code for the Intelligent Coaching Agent (ICA). The InBoxObj directory stores code for the InBox part of the project and so on. The file structure uses some primary object references as file directories. For example, the IcaObj directory is a component that contains primary objects for the ICA such as functional forms, modules and classes. The **BrowserObj** directory contains modules, classes and forms related to the browser functionality in the architecture. The **HTMLGlossary** directory contains code that is used for the HTML reference and glossary component of the architecture. The **IcaObj** directory contains ICA functional code to be used in an application. This code is instantiated and enhanced in accordance with a preferred embodiment. The **InBoxObj** directory contains code pertaining to the inbox functionality used within the architecture. Specifically, there are two major components in this architecture directory. There is a new .ocx control that was created to provide functionality for an inbox in the application. There is also code that provides support for a legacy inbox application. The **PracticeObj** directory contains code for the topics component of the architecture. The topics component can be implemented with the HTMLGlossary component as well. The **QmediaObj** directory contains the components that are media related. An example is the QVIDctrl.cls. The QVIDctrl is the code that creates the links between QVID files in an application and the system in accordance with a preferred embodiment. The **SimObj** directory contains the Simulation Engine, a component of the application that notifies the tutor of inputs and outputs using a spreadsheet to facilitate communication. The **StaticObj** directory holds any component that the application will use statically from the rest of the application. For example, the login form is kept in this folder and is used as a static object in accordance with a preferred embodiment. The **SysDynObj** directory contains the code that allows the Systems Dynamics Engine (Powersim) to pass values to the Simulation Engine and return the values to the tutor. The **VBObj** directory contains common Visual Basic objects used in applications. For example the NowWhat, Visual Basic Reference forms, and specific message box components are stored in this folder. The **_Tools** directory contains two main directories. They represent the two most used tools in accordance with a preferred embodiment. The two directories provide the code for the tools themselves. The reason for providing the code for these tools is to allow a developer to enhance certain parts of the tools to extend their ability. This is important for the current project development and also for the growth of the tools. The **Icautils** directory contains a **data, database, default, graphics, icadoc,** and **testdata** directory. The purpose of all of these directories is to provide a secondary working directory for a developer to keep their testing environment of enhanced Icautils applications separate from the project application. It is built as a testbed for the tool only. No application specific work should be done here. The purpose of each of these directories will be explained in more depth in the project directory section. The TestData folder is unique to the _Tools/ICAUtils directory. It contains test data for the regression bench among others components in ICAUtils.

The Utilities directory holds the available utilities that a Business Simulation project requires for optimal results. This is a repository for code and executable utilities that developers and designers may utilize and enhance in accordance with a preferred embodiment. Most of the utilities are small applications or tools that can be used in the production of

with Target objects (just like the Source to SourceItem relationship). In the journalization example, there is one journal entry for each of the twenty-two transactions. For each journal entry there is a corresponding TargetPage object that contains the Debits Target and Credits Target for that journal entry.

As the student manipulates the application interface, each action is reported to the ICAT. In order to tell the ICAT what actions were taken, the application calls to a database and asks for a specific interface control's ID. When the application has the ID of the target control and the SourceItem control, the application notifies the ICAT about the Target to SourceItem mapping. In other words, every time a student manipulates a source item and associates it with a target (e.g., dragging an account name to a debit line in the journal), the user action is recorded as a mapping of the source item to the target. This mapping is called a UserSourceItemTarget. Figure **21** illustrates the mapping of a source item to a target item in accordance with a preferred embodiment. When the student is ready, he submits his work to one of the simulated team members by clicking on the team member's icon. When the ICAT receives the student's work, it calculates how much of the work is correct by concept. Concepts in our journalization activity will include Debits, Credits, Asset Accounts, etc. For each of these concepts, the ICAT will review all student actions and determine how many of the student actions were correct. In order for the ICAT to understand which targets on the interface are associated with each concept, the targets are bundled into target groups and prioritized in a hierarchy. Once all possible coach topics are activated, a feedback selection analyzes the active pieces of remediation within the concept hierarchy and selects the most appropriate for delivery. The selected pieces of feedback are then assembled into a cohesive paragraph of feedback and delivered to the student. Figure **23** illustrates a feedback selection in accordance with a preferred embodiment. After the ICAT has activated CoachTopics via Rule firings, the Feedback Selection Algorithm is used to determine the most appropriate set of CoachItems (specific pieces of feedback text associated with a CoachTopic) to deliver. The Algorithm accomplishes this by analyzing the concept hierarchy (TargetGroup tree), the active CoachTopics, and the usage history of the CoachItems. Figure **24** is a flowchart of the feedback logic in accordance with a preferred embodiment. There are five main areas to the feedback logic which execute sequentially as listed below. First, the algorithm looks through the target groups and looks for the top-most target group with an active coach topic in it. Second, the algorithm then looks to see if that top-most coach item is praise feedback. If it is praise feedback, then the student has correctly completed the business deliverable and the ICAT will stop and return that coach item. Third, if the feedback is not Praise, then the ICAT will look to see if it is redirect, polish, mastermind or incomplete-stop. If it is any of these, then the algorithm will stop and return that feedback to the user. Fourth, if the feedback is focus, then the algorithm looks to the children target groups and groups any active feedback in these target groups with the focus group header. Fifth, once the feedback has been gathered, then the substitution language is run which replaces substitution variables with the proper names. Once the ICAT has chosen the pieces of feedback to return, the feedback pieces are assembled into a paragraph. With the paragraph assembled, the ICAT goes through and replaces all variables. There are specific variables for SourceItems and Targets. Variables give feedback specificity. The feedback can point out which wrong SourceItems were placed on which Targets. It also provides hints by providing one or two SourceItems which are mapped to the Target.

The Steps Involved in Creating Feedback in Accordance With A Preferred Embodiment

The goal of feedback is to help a student complete a business deliverable. The tutor needs to identify which concepts the student understands and which he does not. The tutor needs to tell the student about his problems and help him understand the concepts. There are seven major steps involved in developing feedback for an application. First, creating a strategy — The designer defines what the student should know. Second, limit errors through interface — The designer

from the simulation model; invoking the RunInputs, RunOutputs and RunLists methods of the simulation engine object in order to bring the ICA to it's original state; calling the Submit method of the ICA object with zero as argument to trigger all the rules; calling the SetDirtyFlag of the ICA object with 0 and false as arguments in order to reset the user's session. Running inputs involves going through the list of TutorAware inputs and notifying the ICA of the SourceItemID, TargetID and Attribute value of every input. Running lists involves going through the list of TutorAware lists and notifying the ICA of the SourceItemID, TargetID and Attribute value of every item in every list. The TargetID is unique for every item in a list. Running outputs involves going through the list of TutorAware outputs and notifying the ICA of the SourceItemID, TargetID and Attribute value of every output. Modification stage 1. **Reading inputs & outputs;** Code: Dim sDataArray(2) as string; Dim vAttribute as variant; Dim lSourceItemID as long; Dim lTargetID as long; lRet = moSimEngine.ReadReference("Distinct_Input", vAttribute, lSourceItemID, lTargetID, sDataArray)

Description: The ReadReference method of the simulation object will return the attribute value of the input or output referenced by name and optionally retrieve the SourceItemID, TargetID and related data. In the current example, the attribute value, the SourceItemID, the TargetID and 3 data cells will be retrieved for the input named Distinct_Input.

Description: The simulation engine object provides basic functionality to manipulate lists.

The ListAdd method appends an item(SourceItemID, Attribute, Data array) to the list. Let's explain the algorithm. First, we find the top of the list using the list name. Then, we seek the first blank cell underneath the top cell. Once the destination is determine, the data is written to the appropriate cells and the ICA is notified of the change.The ListCount method returns the number of items in the specified list. The algorithm works exactly like the ListAdd method but returns the total number of items instead of inserting another element.The ListModify method replaces the specified item with the provided data. Let's explain the algorithm. First, we find the top of the list using the list name. Second, we calculate the row offset based on the item number specified. Then, the ICA is notified of the removal of the existing item. Finally, the data related to the new item is written to the appropriate cells and the ICA is notified of the change. The ListDelete method removes the specified item. The algorithm works exactly like the ListModify method but no new data is added and the cells (width of the list set by 'Total Columns') are deleted with the 'move cells up' parameter set to true. Keep this in mind, as designers often enter the wrong number of columns in the Total Columns parameter. When they overestimate the Total Columns, ListDelete will modify portions of the neighboring list, which leads to erratic behavior when that list is displayed.

SYSTEM DYNAMICS IN ACCORDANCE WITH A PREFERRED EMBODIMENT

To use system dynamics models in the architecture, an engine had to be created that would translate student work into parameters for these models. A complex system dynamics model to interact with an existing simulation architecture is discussed below. The system dynamics model provides the following capabilities.Allow designers to build and test their system dynamics models and ICA feedback before the real interface is built.Reduce the programming complexity of the activities.Centralize the interactions with the system dynamics models.System Dynamics Engine As with the simulation engine, the designer models the task that he/she wants a student to accomplish using a Microsoft Excel spreadsheet. Here, however, the designer also creates a system dynamics model (described later). The system dynamics engine will read all of the significant cells within the simulation model (Excel) and pass these values to the system dynamics model and the ICA. After the system dynamics model runs the information, the output values are read by the engine and then passed to the simulation model and the ICA.

ModelDesc	String*5 0	Description of the model (informational purposes)
SysDynModel	String*5 0	Filename of the actual system dynamics model
Start	Long	Start time to play modal
Stop	Long	Stop time to play model
Step	Long	Interval at which to play one model step and record data

This information is stored in the model table of the simulation database (ICASim.mdb). All of the values that will need to be manually entered by the student that are passed into the system dynamics model are configured as parameter inputs (PInputs) objects. Every PInput has an interface as detailed below.

Field Name	Data Type	Description
PinputID	long	Primary Key for the table
TaskID	long	TaskID of the task associated with the parameter input
ModelID	long	ID of the model associated with the parameter input
InputName	string*50	Name of the parameter input (informational purposes)
InputDesc	string*255	Description (informational purposes)
ReferenceName	string*50	Name of the spreadsheet cell associated with the parameter input
SimReferenceName	string*50	Name of the associated parameter in the system dynamics model
TutorAware	boolean	Whether the ICA should be notified of any input CHANGES
SourceItemID	ong	SourceItemID of the parameter input
TargetID	long	TargetID of the parameter input
Row	long	Spreadsheet row number of the parameter input
Column	long	Spreadsheet column number of the parameter input
SheetName	string*50	Sheet name were the parameter input is located

All of this information is stored for every parameter input in the PInput table of the simulation database (ICASim.mdb). PInputs consist of one spreadsheet cell that can be populated by a designer at design time or by the GBS application at run time via the system dynamics engine object's methods. The purpose of the cell is to provide an entry point to the simulation and system dynamics models. An example of an entry point would be the interest rate parameter in the interest calculation example. The ICA is notified of any changes to the cell when an appropriate activity transpires. When the ICA is notified of a change two messages are sent to the ICA. The first is an ICANotifyDestroy message with the parameter input information i.e., SourceItemID, TargetID and null as an attribute. This message is sent to inform the ICA to remove information from its memory. The second message is an ICANotifyCreate message with the parameter input information i.e., SourceItemID, TargetID, Attribute (cell numeric value). This message advises the ICA to add this information to its memory. A PInput table record in accordance with a preferred embodiment is presented below.

PinputID:	12345
TaskID:	123
ModelID:	1
InputName:	Interest Rate input